(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 4 148 706 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
14.05.2025 Bulletin 2025/20

(21) Application number: **22184268.5**

(22) Date of filing: **12.07.2022**

(51) International Patent Classification (IPC):
G08G 5/06 *(2006.01)* ***G01C 21/20*** *(2006.01)*

(52) Cooperative Patent Classification (CPC):
G01C 21/20; G08G 5/26; G08G 5/34; G08G 5/53; G08G 5/55; G08G 5/723; G08G 5/80;
B64U 2201/10; G08G 5/57

(54) **FAST PATH PLANNING FOR DYNAMIC AVOIDANCE IN PARTIALLY KNOWN ENVIRONMENTS**

SCHNELLE BAHNPLANUNG ZUM DYNAMISCHEN AUSWEICHEN IN TEILWEISE BEKANNTEN UMGEBUNGEN

PLANIFICATION DU TRAJET RAPIDE DESTINÉE À L'ÉVITEMENT DYNAMIQUE DANS DES ENVIRONNEMENTS PARTIELLEMENT CONNU

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **13.09.2021 US 202117447540**

(43) Date of publication of application:
15.03.2023 Bulletin 2023/11

(73) Proprietor: **Aurora Flight Sciences Corporation, a subsidiary of The Boeing Company**
Manassas VA 20110 (US)

(72) Inventors:
- **Bonnoit, Craig John**
  Manassas, 20110 (US)
- **Saunders, Jeffery**
  Manassas, 20110 (US)

(74) Representative: **Witte, Weller & Partner Patentanwälte mbB**
Postfach 10 54 62
70047 Stuttgart (DE)

(56) References cited:
US-A1- 2009 210 109
US-A1- 2016 210 863
US-A1- 2021 134 167

EP 4 148 706 B1

Description

FIELD

[0001] The subject matter described herein generally relates to path planning for a vehicle such as an aircraft.

BACKGROUND

[0002] One of the core capabilities of a computer controlled vehicle is path planning, which specifies a configuration of the vehicle over space and time. Such a plan can then be converted into commands to the vehicle's actuation and ultimately tracked and realized in the physical world (subject to uncertainty). The path planning problem is formalized by definition of goals characterizing what an operator wants the vehicle to do and dynamic feasibility characterizing what the vehicle can do. These components may be designed to incorporate constraints as well. The constraints can include physical limitations, operational rules, and safety considerations. Applications of the path planning problem include computing paths to a goal that avoid collision with obstacles of known extent and are minimal with respect to time, distance, or risk. This is itself a NP-complete problem if the system dynamics pose a holonomic constraint.

[0003] The current state of the art for path planning relies on rapidly exploring random algorithms. For vehicles in dynamic environments, these algorithms operate on an online basis in soft real time and create a log-sparse graph corresponding to feasible paths that a vehicle can take through space. The planning problem is then posed as search over the space of such graphs, and state of the art algorithms make use of dynamic programming to efficiently determine optimal paths. However, the graph creation step is computationally limiting in practice, as generating large graphs in this manner requires an excessive amount of time. Accordingly, only small and shallow graphs are created, which can result in path planning algorithms that fail to converge to near-optimal solutions and lead to abnormal and unacceptable system behavior.

[0004] US 2021/134167 A1 relates to, according to its abstract, systems and methods for autonomous taxi route planning for an aircraft. Clearance communication is received from a ground control station. A planning problem is generated from the clearance communication and sent to a route planner. The route planner receives the planning problem and plans an executable taxi route. Planning the executable taxi route can include generating a complete breadth first search graph from a start pose to a destination, pruning the graph, minimizing the graph, refining the graph, and extracting the shortest path from the graph.

[0005] US 2009/210109 A1, according to its abstract, provides tools and techniques for computing flight plans for unmanned aerial vehicles (UAVs) while routing around obstacles having spatial and temporal dimensions. Methods provided by these tools may receive data representing destinations to be visited by the UAVs, and may receive data representing obstacles having spatial and temporal dimensions. These methods may also calculate trajectories spatial and temporal dimensions, by which the UAV may travel from one destination to another, and may at least attempt to compute flight plans for the UAVs that incorporate these trajectories. The methods may also determine whether these trajectories intersect any obstacles, and at least attempt to reroute the trajectories around the obstacles. These tools may also provide systems and computer-readable media containing software for performing any of the foregoing methods.

SUMMARY

[0006] The solution is provided by the features of the independent claims. Variations are as described by the features of the dependent claims.

[0007] According to various examples, a method of traversing in an environment that includes at least one obstacle, by a mobile autonomous system, to a destination in the environment, is presented. The method includes generating, prior to the mobile autonomous system commencing activity in the environment, a graph comprising a plurality of vertices representing positions in the environment and a plurality of edges between vertices representing feasible transitions by the mobile autonomous vehicle in the environment; annotating the graph with at least one edge connecting a representation of a present position of the mobile autonomous system to a vertex of the graph; determining, based on the graph, a path from the present position of the mobile autonomous system in the environment to the destination; and traversing the environment to the destination, by the mobile autonomous system, based on the path.

[0008] Various optional features of the above method include the following. The vertices may store location and direction information. The vertices may store respective costs to the destination. The vertices may store identifications of next vertices in traversing to the destination. The at least one obstacle may include at least one dynamic obstacle. The method may further include invalidating at least one edge in the graph to represent a position of the at least one obstacle at a particular time. The annotating the graph with the at least one edge may include adding to the graph the at least one edge without generating a new memory allocation on an order of a size of the graph. The mobile autonomous system may include an aircraft. The destination may include a plurality of locations, and the graph may represent the plurality of locations by a plurality of vertices. The determining the path may include performing a non-exhaustive search of the graph.

[0009] According to various examples, a system for traversing in an environment that includes at least one obstacle, by a mobile autonomous system, to a destina-

tion in the environment, is presented. The system includes an electronic processor; electronic persistent memory comprising instructions that, when executed by the electronic processor, configure the electronic processor to perform operations comprising: generating, prior to the mobile autonomous system commencing activity in the environment, a graph comprising a plurality of vertices representing positions in the environment and a plurality of edges between vertices representing feasible transitions by the mobile autonomous vehicle in the environment; annotating the graph with at least one edge connecting a representation of a present position of the mobile autonomous system to a vertex of the graph; and determining, based on the graph, a path from the present position of the mobile autonomous system in the environment to the destination; wherein the mobile autonomous system is configured to traverse the environment to the destination based on the path.

[0010] Various optional features of the above system include the following. The vertices may store location and direction information. The vertices may store respective costs to the destination. The vertices may store identifications of next vertices in traversing to the destination. The at least one obstacle may include at least one dynamic obstacle. The operations may further include invalidating at least one edge in the graph to represent a position of the at least one obstacle at a particular time. The annotating the graph with the at least one edge may include adding to the graph the at least one edge without generating a new memory allocation on an order of a size of the graph. The mobile autonomous system may include an aircraft. The destination may include a plurality of locations, and the graph may represent the plurality of locations by a plurality of vertices. The determining the path may include performing a non-exhaustive search of the graph.

DRAWINGS

[0011] The above and/or other aspects and advantages will become more apparent and more readily appreciated from the following detailed description of examples, taken in conjunction with the accompanying drawings, in which:

Fig. 1 depicts an example offline graph according to various examples;

Fig. 2 depicts the example augmented offline graph of Fig. 1, including a minimal path from a starting position to a destination according to various examples;

Fig. 3 depicts an example online graph according to various examples;

Fig. 4 depicts an online graph, representing the example online graph of Fig. 3, and showing a minimal path from the current location of the vehicle to the destination, according to various examples;

Fig. 5 is a block diagram of a path planning system according to various examples;

Fig. 6 is a flowchart for a method of path planning according to various examples; and

Fig. 7 is a block diagram of a example hardware for implementing various examples.

DETAILED DESCRIPTION

[0012] Exemplary aspects will now be described more fully with reference to the accompanying drawings. Examples of the disclosure, however, can be embodied in many different forms and should not be construed as being limited to the examples set forth herein. Rather, these examples are provided so that this disclosure will be thorough and complete, and will fully convey the scope to those skilled in the art. In the drawings, some details may be simplified and/or may be drawn to facilitate understanding rather than to maintain strict structural accuracy, detail, and/or scale.

[0013] Some examples overcome the computation time and graph size bottlenecks of existing path planning techniques. Some examples are particularly suited for problem instances with a mixed structure that includes both known static hazards and dynamically observed hazards. In such cases, some examples allow for the use of graphs that are orders of magnitude larger (for equivalent computational work) than what are currently used. This allows examples to quickly compute solutions that are both locally and globally performant. Conversely, for a fixed requirement on expressive capacity, some examples provide improved runtime and faster closing of control loop. This allows a vehicle to more readily adapt to new information without losing sight of long-run goals.

[0014] Examples may be utilized with a variety of vehicles in a variety of environments. Environments may be three-dimensional, e.g., for airspace or underwater path planning, or two-dimensional, e.g., for ground or sea surface path planning. Example vehicles include aircraft, ships, submarines, automobiles, trucks, factory robots, unmanned ariel, submersible, and terrestrial vehicles, and any other vehicle that is autonomous or that can be controlled by a computer autonomously or semi-autonomously.

[0015] Some examples are particularly relevant for scenarios in which the implications of local maneuvering affect long-run mission feasibility. One example of this sort of mixed objective is the safe maneuvering of an energetically constrained vehicle such as an electric vertical takeoff and landing (eVTOL) airplane around hazards such as birds or other aircraft, in which case proximate collision avoidance must be balanced against long run energetic limitations necessary to safely land

while avoiding known terrain and hazards. A second example is a strongly underactuated vehicle such as a high-altitude glider, which might need to balance dynamism to harness energy from locally observed wind gusts with station keeping to a sufficient degree for long run mission success.

[0016] In general, examples may include an offline portion and an online portion. The offline portion may be executed prior to the vehicle operating in the environment under consideration, and the online portion may be executed during the vehicle's operation in the environment under consideration. The offline portion can include generating an offline graph that represents relations between reachable configurations that avoid static and otherwise known obstacles and satisfy operational constraints. An example offline graph is depicted in Fig. 1. The offline portion can further include augmenting the offline graph with vertex-wise information on the minimal cost paths and next vertex information. An example such augmented offline graph is depicted in Fig. 2. The online portion can include generating an online graph that includes at least one edge connecting a current position of the vehicle to a vertex in the offline graph. An example such an online graph is depicted in Fig. 3. The online portion can also include augmenting the online graph to evaluate if an edge is blocked by one or more previously unobserved obstacles. An example such an augmented online graph is depicted in Fig. 3. The online portion can further include searching for an updated minimal path subject to the one or more previously unobserved obstacles. The results of an example such search are depicted in Fig. 4.

[0017] These and other elements, features, and advantages are shown and described presently in reference to Figs. 1-7.

[0018] Fig. 1 depicts an example offline graph 100 according to various examples. Offline graph 100 may include a representation of a known starting location and/or a known destination, e.g., by way of non-limiting examples, an origin airport and/or a destination airport. More generally, offline graph is populated with vertices, e.g., vertex 102, that each represent a state that includes both location and direction.

[0019] The points used to seed offline graph 100, e.g., the points that are to be used for the location portion of the vertices, can be selected according to any of a variety of techniques. By way of non-limiting examples, such points may be selected randomly, by using a deterministic space filling algorithm, or to align with existing procedural or operational structures (e.g., known fixes, airways, or approaches in an airspace system). Offline graph 100 may include, for example, at least 100,000 vertices.

[0020] Each vertex in the graph includes both positional and velocity degrees of freedom, representing location and direction with speed, respectively. Various examples may use any mixture of such states, e.g., position, heading, and airspeed for an aircraft. Although offline graph 100 represents two-dimensional space, examples are not so limited. Offline graphs (and other graphs disclosed herein) representing three-dimensional space are contemplated. For example, in three dimensions, each vertex may include two three-tuples, e.g., (x_1, y_1, z_1), (x_2, y_2, z_2), with one representing position and another representing orientation. Thus, for three dimensions, each vertex may include two three-dimensional vectors. In two dimensions, each vertex may include two pairs, e.g., (x_1, y_1), (x_2, y_2), with one representing position and another representing orientation; in two dimensions, each vertex may include two two-dimensional vectors. In sum, each vertex in offline graph 100 includes both a location and a direction. When clear from context, vertices may be referred to as states, which includes both location and direction, or as locations, which refers to the location portion. Whether two-dimensional or three-dimensional, offline graph 100 may include an added dimension (e.g., added to each of the respective three-tuples or pairs) to represent time.

[0021] Edges between vertices in offline graph 100, e.g., edge 104, represent feasible transitions between states (e.g., locations and directions) respecting known constraints. Feasible transitions can take into account, for example physical limitations of the vehicle, such as turning radius, and constraints can take into account known obstacles. In general, offline graph 100 may be a log sparse graph, with |EDGES| = |VERTICES| log|VERTICES|.

[0022] Offline graph 100 may be generated offline, prior to the vehicle operating in the space represented by offline graph 100. Offline graph 100 may be stored electronically in persistent computer-readable media, for example. In particular, offline graph 100, as well as the other graphs disclosed herein, may be stored in any suitable graph storage data structure, modified to store various data as disclosed herein, e.g., shortest path and successor vertex data as described in reference to Fig. 2, in association with the vertices.

[0023] Fig. 2 depicts an example augmented offline graph 200, in particular, an augmentation of offline graph 100 of Fig. 1, including a minimal path 206 from a starting location 202 to a destination location 204, according to various examples. Starting location 202 and destination location 204 may each be represented by a respective vertex in augmented offline graph 200, indicating both location and direction for each. Each vertex in augmented offline graph 200 further includes a representation of a cost to destination state 203, e.g., a minimal cost. Thus, each vertex may be augmented with a scalar value representing such a cost. Herein, "cost" may be implemented in a variety of forms and may represent time, distance, fuel expense, or any combination thereof. Further, each vertex may be augmented with an identification of a successor vertex in a path, e.g., a minimal cost path, to destination state 204. The identification of the successor vertex may take any of a variety of forms, e.g., an index of a vertex, where the vertices of augmented offline graph 200 are indexed by some enumeration,

for example.

[0024] As shown, augmented offline graph 200 includes minimal cost path 206 from starting state 202 to destination state 204. Such a path may be obtained using a variety of techniques, e.g., dynamic programming applied to the costs to the destination stored at the vertices. Example suitable dynamic programming techniques include Dijkstra's algorithm and A*.

[0025] Fig. 3 depicts an example online graph 300 according to various examples. Online graph 300 may be constructed from an augmented offline graph, and is described by way of non-limiting example in reference to augmented offline graph 200 of Fig. 2. Relative to the offline portion of various examples, for the online portion, when the vehicle is present in the environment under consideration, two things may change. First, the current location 302 of the vehicle may be at an arbitrary position, e.g., not at a location represented by a vertex of augmented offline graph 200. Second, one or more previously unknown obstacles, such as obstacle 306, may be present. These changes may be addressed as follows. First, new edges 308 that connect the current location 302 of the vehicle may be added. Second, edges 310 that cross obstacle 306 may be considered invalid. Such edges may not be considered when planning a path, but may be retained in the graph, e.g., for use when the obstacle is no longer present.

[0026] The current location 302 of the vehicle may be determined according to any of a variety of techniques, including, by way of non-limiting examples, GPS, satellite imaging, dead reckoning, and/or triangulation based on any of RADAR, SONAR, or LIDAR. The current location and direction form a state that may be joined to an existing vertex in online graph 300, where the existing vertex may be identified, for example, using a nearest neighbor search.

[0027] The new obstacles, such as obstacle 306, may be identified using any of a variety of techniques. Detection techniques include, by way of non-limiting example, ground-based, air-based, sea-based, and/or vehicle-based RADAR, SONAR, and LIDAR. Satellite imaging may be used. Information from any entity, such as an air traffic controller, that may utilize any of the aforementioned detection techniques or a different technique, may further be used to identify any new obstacles that appear in the environment under consideration.

[0028] Online graph 300 may include an added dimension to represent time. For example, for path planning in three dimensions, each vertex may include a three-dimensional location vector, a three-dimensional direction vector, and a scalar time value, all of which may be represented in seven dimensions, e.g., in $\mathbb{R}^7$. Further, each edge may be associated with a traversal duration. When planning in the presence of dynamic obstacles, time as a parameter may be used to evaluate if there is a potential collision. For example, if the current time is T and an edge in the graph is known to have duration D to traverse, then the state that may be reached by following that edge is ([location vector], [direction vector], *T+D*), which allows for an evaluation of whether there is a potential collision with the dynamic obstacle. The graph may be constructed lazily as-needed (with caching to prevent recalculation).

[0029] Fig. 4 depicts an online graph 400, representing the example online graph 300 of Fig. 3, and showing a minimal cost path 402 from the current location 302 of the vehicle to the destination 204, according to various examples. Minimal path 402 may be obtained using any of a variety of search algorithms applied to online graph 400, e.g., beam search. In particular, the search may use the cost to destination 203 stored at the vertices as a proximate metric to select the minimal cost path 402.

[0030] In sum, as shown and described herein in reference to Figs. 1-4, a technique for path planning is presented. The technique includes an offline portion, prior to commencing vehicle operations in the area under consideration, in which an offline graph is generated. The offline graph may be implemented as a large log-sparse mobility graph representing relations between reachable configurations that avoid static and known obstacles and satisfy operational constraints. Fig. 1 depicts an example such offline graph. The offline portion may further include augmenting the offline graph with vertex-wise information on the minimal cost paths to the destination, e.g., by using dynamic programming. Fig. 2 depicts an example such augmented offline graph. The technique also includes an online portion, during vehicle activity in the area under consideration, that uses the augmented offline graph to generate an online graph. The online graph includes a connection of a current position of the vehicle to a vertex in the augmented offline graph. Fig. 3 depicts an example online graph with such a connection. The online graph further includes the ability to evaluate if an edge is blocked by any previously unobserved obstacle. Fig. 3 depicts an example online graph with such an ability. The online graph is then used to plan a path, e.g., by searching for a minimal path subject to previously unobserved obstacles. Fig. 4 depicts an example minimal path. Fig. 5, described presently, shows a system configured to perform the described technique.

[0031] Fig. 5 is a block diagram of a path planning system 500 according to various examples. System 500 divides the work required for path planning into actions performed by offline portion 510, which includes actions that may occur prior to vehicle operation in the area of interest, and actions performed by online 520 portion, which includes actions that may be performed during vehicle operation in the area of interest. In general, offline portion 510 generates an offline graph as a computational representation of mobility in a specific operational context. This can be done long before use is required to solve a specific planning problem from the context of a vehicle, and admits use of hardware parallelism and validation approaches that would not be otherwise computationally tractable in the runtime of planning.

Online portion 520 generally concerns the enrichment and consumption of the offline graph within the area of operation during operation.

[0032] Offline portion 510 includes build offline graph component 512. Build offline graph component 512 provides an offline graph representing feasible connections between a set of states and a set of goals, subject to known dynamics and static time-invariant constraints. Build offline graph component 512 may augment each vertex of the offline graph with a minimal cost to go to the destination. Persisting such cost-to-go information over each vertex allows for the use of purely local O(1) greedy heuristics about reachability. Further, build offline graph component 512 may augment the offline graph with geometric information in each vertex representing path connections, e.g., a next vertex identification, to improve the speed at which an edge can be evaluated as intersecting a known obstacle. Once constructed by build offline graph component 512, the offline graph may be stored in persistent storage 514 for consumption during the online portion.

[0033] System 500 also includes online portion 520. Online portion 520 includes planning scope 530, a runtime environment that supports persistent memory allocation to reuse the offline graph without reloading or reallocating from memory. Planning scope 530 may be used in a repeated context, including receding horizon planning or evaluation of multiple scenarios. Planning scope 530 may retrieve the offline graph from persistent storage 522, which may be the same or a different persistent storage from persistent storage 514.

[0034] Online portion 520 also includes add edge component 524, which adds at least one edge connecting a representation of a current location of the vehicle to a vertex in the offline graph. That is, add edge component 524 provides a graph interface representing the online graph inclusive of an arbitrary current location (e.g., a current state) connected onto the existing offline graph. The connection may be determined using a nearest neighbor search, for example. Add edge component 524 may provide an abstract graph view rather than a new full-size memory allocation in order to avoid memory operations on the order of the size of the graph (e.g., O(|VERTICES|)) during online portion 520. Such an abstract graph view may provide a function that can, for example, identify all successors of an input vertex identification. Such a function may operate on a small portion of the graph without requiring the entire graph to be held in dynamic memory. Further, such a function may store a temporary list of vertices that have previously been evaluated by the function and their successors, such that future evaluations of the function at such vertices may be performed using a fast lookup call.

[0035] Online portion 520 further includes invalidate blocked edge component 526. Invalidate blocked edge component 526 provides a graph interface representing an offline graph inclusive of known dynamic constraints, including moving objects known during the online portion. Invalidate blocked edge component 526 may employ halfspace inclusion or polytope intersection together with next vertex information to detect whether a dynamic or static constraint blocks an edge. If so, blocked edge component 526 removes the edge from being used in computations at later stages. Blocked edge component 526 may keep an electronically stored list of blocked edges in memory according to some examples. Such a list may be checked prior to performing certain actions, such as performed by graph planner 528, and edges represented therein may be removed from the actions. Similar to add edge component 524, invalidate blocked edge component 526 may provide an abstract graph view rather than a new memory allocation in order to avoid memory operations on the order of the size of the graph during online portion 520.

[0036] Online portion 520 further includes graph planner 528. Graph planner 528 provides trajectories, paths, or plans, given goals. Graph planner 528 may implement any of a variety of algorithms, e.g., bounding (A*, D*) random search (e.g., Monte Carlo tree search), or heuristic search algorithms (e.g., beam search). Because the removing blocked edges from being used can increase the cost-to-go for vertices in a non-local manner, graph planner 528 may employ backtracking heuristic modifications. According to some examples, a beam search heuristic is used to balance runtime and optimality considerations. In particular, a logarithmically scaling beam width may be used, e.g., the beam width may be selected as c log|VERTICES|, where c is some constant. Graph planner 528 outputs a path from the current location of the vehicle to the destination.

[0037] Some examples minimize the amount of runtime work performed by online portion 520 through the use of various features. Such features may include (1) creating abstract view graphs instead of reallocating memory, and (2) updating cost-to-go lazily based on backtracking instead of exhaustively checking edges and performing computational work that scales extensively with number of edges. These two aspects in particular may provide performant and accurate planning with relatively low requirements on dynamic memory allocation and computational complexity at runtime.

[0038] Fig. 6 is a flowchart for a method 600 of path planning according to various examples. Method 600 may include a mobile autonomous system traversing an environment that includes at least one obstacle to a destination in the environment. Method 600 may be implemented by system 500 as shown and described in reference to Fig. 5 using hardware 700 as shown and described in reference to Fig. 7.

[0039] At 602, method 600 generates an offline graph. The offline graph may be generated prior to the mobile autonomous system commencing activity in the environment. The offline graph may include a plurality of vertices representing positions in the environment and a plurality of edges between vertices representing feasible transitions by the mobile autonomous vehicle in the environ-

ment. The actions of 602 may include any, or any combination, of actions as shown and described herein in reference to Figs. 1 and 2 and in reference to build offline graph component 512 of Fig. 5.

[0040] At 604, method 600 includes annotating the graph with at least one edge connecting a representation of a present position of the mobile autonomous system to a vertex of the graph. The actions of 604 may include any, or any combination, of actions as shown and described in reference to Fig. 3 and in reference to add edge component 524 of Fig. 5.

[0041] At 606, method 600 includes determining, based on the graph, a path from the present position of the mobile autonomous system in the environment to the destination. The actions of 606 may include any, or any combination, of actions as shown and described in reference to Fig. 4 and in reference to graph planner 528 of Fig. 5.

[0042] At 608, method 600 includes traversing the environment to the destination, by the mobile autonomous system, based on the path. The actions of 608 may include the mobile autonomous system physically traversing in the environment to the destination using the path as a traversal route.

[0043] Fig. 7 is a block diagram of example hardware 700 for implementing various examples. For example, Fig. 7 illustrates various hardware, software, and other resources that can be used in implementations of method 600 as shown and described herein in reference to Fig. 6. Further, system 700 may implement build offline graph component 512, persistent storage 514, persistent storage 522, add edge component 524, invalidate blocked edge component 526, graph planner 528, and/or planning scope 530.

[0044] System 700 includes offline portion computer 720 and online portion computer 710. Offline portion computer 720 may perform offline portion actions, e.g., as shown and described herein in reference to Figs. 1, 2, and 5. Online portion computer 710 may perform online portion actions, e.g., as shown and described herein in reference to Figs. 3, 4, and 5. Online portion computer 710 may be deployed aboard the vehicle at issue, e.g., aircraft 702. Offline portion computer 710 and online portion computer 720 may be communicatively coupled by way of one or more networks 730, e.g., the internet.

[0045] Either of offline portion computer 720 or online portion computer 710 may be implemented as any of a desktop computer, a laptop computer, can be incorporated in one or more servers, clusters, or other computers or hardware resources, or can be implemented using cloud-based resources. Offline portion computer 720 includes volatile memory 726 and persistent memory 728, the latter of which can store computer-readable instructions, that, when executed by electronic processor 722, configure offline portion computer 720 to at least partially perform an offline portion of methods, e.g., method 600, as shown and described herein. Offline portion computer 720 includes network interface 724, which communicatively couples offline portion computer 720 to online portion computer 710 via network 730. Online computer 710 includes volatile memory 716 and persistent memory 718, the latter of which can store computer-readable instructions, that, when executed by electronic processor 712, configure online computer 710 to at least partially perform an online portion of methods, e.g., method 600, as shown and described herein. Online portion computer 710 includes network interface 714, which communicatively couples online portion computer 710 to offline portion computer 720 via network 730. Other configurations of system 700, associated network connections, and other hardware, software, and service resources are possible.

[0046] Many variations and modifications of the disclosed example examples are possible. According to some examples, the destination state could be replaced with a union of states, e.g., to represent dynamically choosing amongst several feasible landing locations or a range of admissible finishing states (such as in a holding pattern, in a declared safe landing zone, or at an airport). Note that the cost-to-go metric may remain scalar valued, assuming that the goals are to be treated as equivalent priority. According to some examples, the maneuverability graph may be replaced with a local template reflecting a sequence of maneuvers in cases for which translational invariance was present (such as for planning maneuvers in a dogfight in up and away flight). In this case, the offline actions may capture representations of how basic controllable actions can be composed into longer run maneuvers that are searched at runtime.

Claims

1. A method (600) of traversing in an environment that includes at least one obstacle (306), by a mobile autonomous system, to a destination (204) in the environment, the method comprising:

   generating, prior to the mobile autonomous system commencing activity in the environment, a graph comprising a plurality of vertices (102) representing positions in the environment and a plurality of edges (104) between vertices representing feasible transitions by the mobile autonomous vehicle in the environment, wherein the graph is a log-sparse graph with |EDGES| = |VERTICES| log |VERTICES|;

   annotating the graph with at least one edge connecting a representation of a present position of the mobile autonomous system to a vertex of the graph;

   determining, based on the graph, a path from the present position of the mobile autonomous system in the environment to the destination; and

   traversing the environment to the destination, by

the mobile autonomous system, based on the path.

2. The method of claim 1, wherein generating the vertices comprises storing for each of the vertices location and direction information.

3. The method of any of claims 1 to 2, wherein generating the vertices comprises storing for each of the vertices respective costs to the destination.

4. The method of any of claims 1 to 3, wherein generating the vertices comprises storing for each of the vertices identifications of next vertices in traversing to the destination.

5. The method of any of claims 1 to 4, wherein the at least one obstacle comprises at least one dynamic obstacle.

6. The method of any of claims 1 to 5, further comprising invalidating at least one edge in the graph to represent a position of the at least one obstacle at a particular time.

7. The method of any of claims 1 to 6, wherein the annotating the graph with the at least one edge comprises adding to the graph the at least one edge without generating a new memory allocation on an order of a size of the graph.

8. The method of any of claims 1 to 7, wherein the destination comprises a plurality of locations, and wherein the graph represents the plurality of locations by a plurality of vertices.

9. The method of any of claims 1 to 8, wherein the determining the path comprises performing a non-exhaustive search of the graph.

10. A system (500, 700) for traversing in an environment that includes at least one obstacle (306), by a mobile autonomous system, to a destination (204) in the environment, the system comprising:

an electronic processor (712, 722); and
electronic persistent memory (718, 728) comprising instructions that, when executed by the electronic processor, configure the electronic processor to perform operations comprising:
generating, prior to the mobile autonomous system commencing activity in the environment, a graph comprising a plurality of vertices representing positions in the environment and a plurality of edges (104) between vertices representing feasible transitions by the mobile autonomous vehicle in the environment,
wherein the graph is a log-sparse graph with | EDGES| = |VERTICES| log | VERTICES|;

annotating the graph with at least one edge connecting a representation of a present position of the mobile autonomous system to a vertex (102) of the graph; and
determining, based on the graph, a path from the present position of the mobile autonomous system in the environment to the destination;

wherein the mobile autonomous system is configured to traverse the environment to the destination based on the path.

11. The system of claim 10, wherein generating the vertices comprises storing for each of the vertices location and direction information.

12. The system of claim 10, wherein generating the vertices comprises storing for each of the vertices respective costs to the destination.

13. The system of claim 10, wherein generating the vertices comprises storing for each of the vertices identifications of next vertices in traversing to the destination.

14. The system of any of claims 10 to 13, wherein the annotating the graph with the at least one edge comprises adding to the graph the at least one edge without generating a new memory allocation on an order of a size of the graph.

15. The system of any of claims 10 to 14, wherein the destination (203, 204) comprises a plurality of locations, and wherein the graph represents the plurality of locations by a plurality of vertices.

Patentansprüche

1. Verfahren (600) zum Durchqueren einer Umgebung, die mindestens ein Hindernis (306) aufweist, durch ein mobiles autonomes System zu einem Zielort (204) in der Umgebung, wobei das Verfahren Folgendes aufweist:

Erzeugen, vor dem Beginn der Aktivität des mobilen autonomen Systems in der Umgebung, eines Graphen, der eine Vielzahl von Knoten (102) aufweist, die Positionen in der Umgebung darstellen, und eine Vielzahl von Kanten (104) zwischen Knoten aufweist, die durch das mobile autonome Fahrzeug in der Umgebung realisierbare Übergänge darstellen, wobei der Graph ein log-sparsamer Graph mit |KANTEN| = |KNOTEN| log |KNOTEN| ist;

Annotieren des Graphen mit mindestens einer Kante, die eine Darstellung einer aktuellen Position des mobilen autonomen Systems mit einem Knoten des Graphen verbindet;
Bestimmen eines Pfads auf der Grundlage des Graphen von der aktuellen Position des mobilen autonomen Systems in der Umgebung zu dem Zielort; und
Durchqueren der Umgebung zu dem Zielort durch das mobile autonome System auf der Grundlage des Pfads.

2. Verfahren nach Anspruch 1, wobei das Erzeugen der Knoten das Speichern von Orts- und Richtungsinformationen für jeden der Knoten aufweist.

3. Verfahren nach einem der Ansprüche 1 bis 2, wobei das Erzeugen der Knoten das Speichern jeweiliger Kosten zum Zielort für jeden der Knoten aufweist.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei das Erzeugen der Knoten das Speichern von Identifikationen nächster Knoten zur Durchquerung zum Zielort für jeden der Knoten aufweist.

5. Verfahren nach einem der Ansprüche 1 bis 4, wobei das mindestens eine Hindernis mindestens ein dynamisches Hindernis aufweist.

6. Verfahren nach einem der Ansprüche 1 bis 5, ferner aufweisend das Ungültigmachen von mindestens einer Kante in dem Graphen zur Darstellung einer Position des mindestens einen Hindernisses zu einem bestimmten Zeitpunkt.

7. Verfahren nach einem der Ansprüche 1 bis 6, wobei das Annotieren des Graphen mit der mindestens einen Kante das Hinzufügen der mindestens einen Kante zum Graphen ohne Erzeugen einer neuen Speicherzuweisung in einer Größenordnung der Größe des Graphen aufweist.

8. Verfahren nach einem der Ansprüche 1 bis 7, wobei der Zielort eine Vielzahl von Orten aufweist, und wobei der Graph die Vielzahl von Orten durch eine Vielzahl von Knoten darstellt.

9. Verfahren nach einem der Ansprüche 1 bis 8, wobei das Bestimmen des Pfads das Durchführen einer nicht erschöpfenden Suche des Graphen aufweist.

10. System (500, 700) zum Durchqueren einer Umgebung, die mindestens ein Hindernis (306) aufweist, durch ein mobiles autonomes System zu einem Zielort (204) in der Umgebung, wobei das System Folgendes aufweist:

einen elektronischen Prozessor (712, 722); und

elektronischen persistenten Speicher (718, 728), der Anweisungen aufweist, die, wenn sie durch den elektronischen Prozessor ausgeführt werden, den elektronischen Prozessor dazu konfigurieren, Operationen auszuführen, die Folgendes aufweisen:

Erzeugen, vor dem Beginn der Aktivität des mobilen autonomen Systems in der Umgebung, eines Graphen, der eine Vielzahl von Knoten aufweist, die Positionen in der Umgebung darstellen, und eine Vielzahl von Kanten (104) zwischen Knoten aufweist, die durch das mobile autonome Fahrzeug in der Umgebung realisierbare Übergänge darstellen, wobei der Graph ein log-sparsamer Graph mit |KANTEN| = |KNOTEN| log | KNOTEN| ist;
Annotieren des Graphen mit mindestens einer Kante, die eine Darstellung einer aktuellen Position des mobilen autonomen Systems mit einem Knoten (102) des Graphen verbindet; und
Bestimmen, auf der Grundlage des Graphen, eines Pfads von der aktuellen Position des mobilen autonomen Systems in der Umgebung zu dem Zielort;

wobei das mobile autonome System dazu konfiguriert ist, die Umgebung zu dem Zielort auf der Grundlage des Pfads zu durchqueren.

11. System nach Anspruch 10, wobei das Erzeugen der Knoten das Speichern von Orts- und Richtungsinformationen für jeden der Knoten aufweist.

12. System nach Anspruch 10, wobei das Erzeugen der Knoten das Speichern jeweiliger Kosten zum Zielort für jeden der Knoten aufweist.

13. System nach Anspruch 10, wobei das Erzeugen der Knoten das Speichern von Identifikationen nächster Knoten zur Durchquerung zum Zielort für jeden der Knoten aufweist.

14. System nach einem der Ansprüche 10 bis 13, wobei das Annotieren des Graphen mit der mindestens einen Kante das Hinzufügen der mindestens einen Kante zum Graphen ohne Erzeugen einer neuen Speicherzuweisung in einer Größenordnung der Größe des Graphen aufweist.

15. System nach einem der Ansprüche 10 bis 14, wobei der Zielort (203, 204) eine Vielzahl von Orten aufweist, und wobei der Graph die Vielzahl von Orten durch eine Vielzahl von Knoten darstellt.

Revendications

1. Procédé (600) de traversée dans un environnement qui comporte au moins un obstacle (306), par un système autonome mobile, vers une destination (204) dans l'environnement, le procédé comprenant :

   la génération, avant que le système mobile autonome ne commence son activité dans l'environnement, d'un graphe comprenant une pluralité de sommets (102) représentant des positions dans l'environnement et une pluralité d'arêtes (104) entre des sommets représentant des transitions réalisables par le véhicule mobile autonome dans l'environnement, dans lequel le graphe est un graphe log peu dense avec | ARÊTES| = |SOMMETS| log ISOMMETSI ;
   l'annotation du graphe avec au moins une arête reliant une représentation d'une position actuelle du système autonome mobile à un sommet du graphe ;
   la détermination, sur la base du graphe, d'un chemin entre la position actuelle du système autonome mobile dans l'environnement et la destination ; et
   la traversée de l'environnement vers la destination, par le système autonome mobile, sur la base du chemin.

2. Procédé selon la revendication 1, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, d'informations d'emplacement et de direction.

3. Procédé selon l'une quelconque des revendications 1 à 2, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, des coûts respectifs vers la destination.

4. Procédé selon l'une quelconque des revendications 1 à 3, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, des identifications des sommets suivants dans la traversée vers la destination.

5. Procédé selon l'une quelconque des revendications 1 à 4, dans lequel l'au moins un obstacle comprend au moins un obstacle dynamique.

6. Procédé selon l'une quelconque des revendications 1 à 5, comprenant en outre l'invalidation d'au moins une arête du graphe pour représenter une position de l'au moins un obstacle à un moment précis.

7. Procédé selon l'une quelconque des revendications 1 à 6, dans lequel l'annotation du graphe avec l'au moins une arête comprend l'ajout au graphe de l'au moins une arête sans générer une nouvelle allocation de mémoire d'un ordre d'une taille du graphe.

8. Procédé selon l'une quelconque des revendications 1 à 7, dans lequel la destination comprend une pluralité d'emplacements, et dans lequel le graphe représente la pluralité d'emplacements par une pluralité de sommets.

9. Procédé selon l'une quelconque des revendications 1 à 8, dans lequel la détermination du chemin comprend la réalisation d'une recherche non exhaustive dans le graphe.

10. Système (500, 700) de traversée dans un environnement comportant au moins un obstacle (306), par un système autonome mobile, vers une destination (204) dans l'environnement, le système comprenant :

    un processeur électronique (712, 722) ; et
    une mémoire électronique persistante (718, 728) comprenant des instructions qui, lorsqu'elles sont exécutées par le processeur électronique, configurent le processeur électronique pour réaliser les opérations comprenant :

    la génération, avant que le système autonome mobile ne commence son activité dans l'environnement, d'un graphe comprenant une pluralité de sommets représentant des positions dans l'environnement et une pluralité d'arêtes (104) entre des sommets représentant des transitions réalisables par le véhicule autonome mobile dans l'environnement, dans lequel le graphe est un graphe log peu dense avec |ARÊTES| = | SOMMETS| log |SOMMETS| ;
    l'annotation du graphe avec au moins une arête reliant une représentation d'une position actuelle du système autonome mobile à un sommet (102) du graphe ; et
    la détermination, sur la base du graphe, d'un chemin entre la position actuelle du système autonome mobile dans l'environnement et la destination ;
    dans lequel le système autonome mobile est configuré pour traverser l'environnement vers la destination sur la base du chemin.

11. Système selon la revendication 10, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, d'informations d'emplacement et de direction.

12. Système selon la revendication 10, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, des coûts respectifs vers

la destination.

13. Système selon la revendication 10, dans lequel la génération des sommets comprend le stockage, pour chacun des sommets, des identifications des sommets suivants dans la traversée vers la destination.

14. Système selon l'une quelconque des revendications 10 à 13, dans lequel l'annotation du graphe avec l'au moins une arête comprend l'ajout au graphe de l'au moins une arête sans générer une nouvelle allocation de mémoire d'un ordre d'une taille du graphe.

15. Système selon l'une quelconque des revendications 10 à 14, dans lequel la destination (203, 204) comprend une pluralité d'emplacements, et dans lequel le graphe représente la pluralité d'emplacements par une pluralité de sommets.

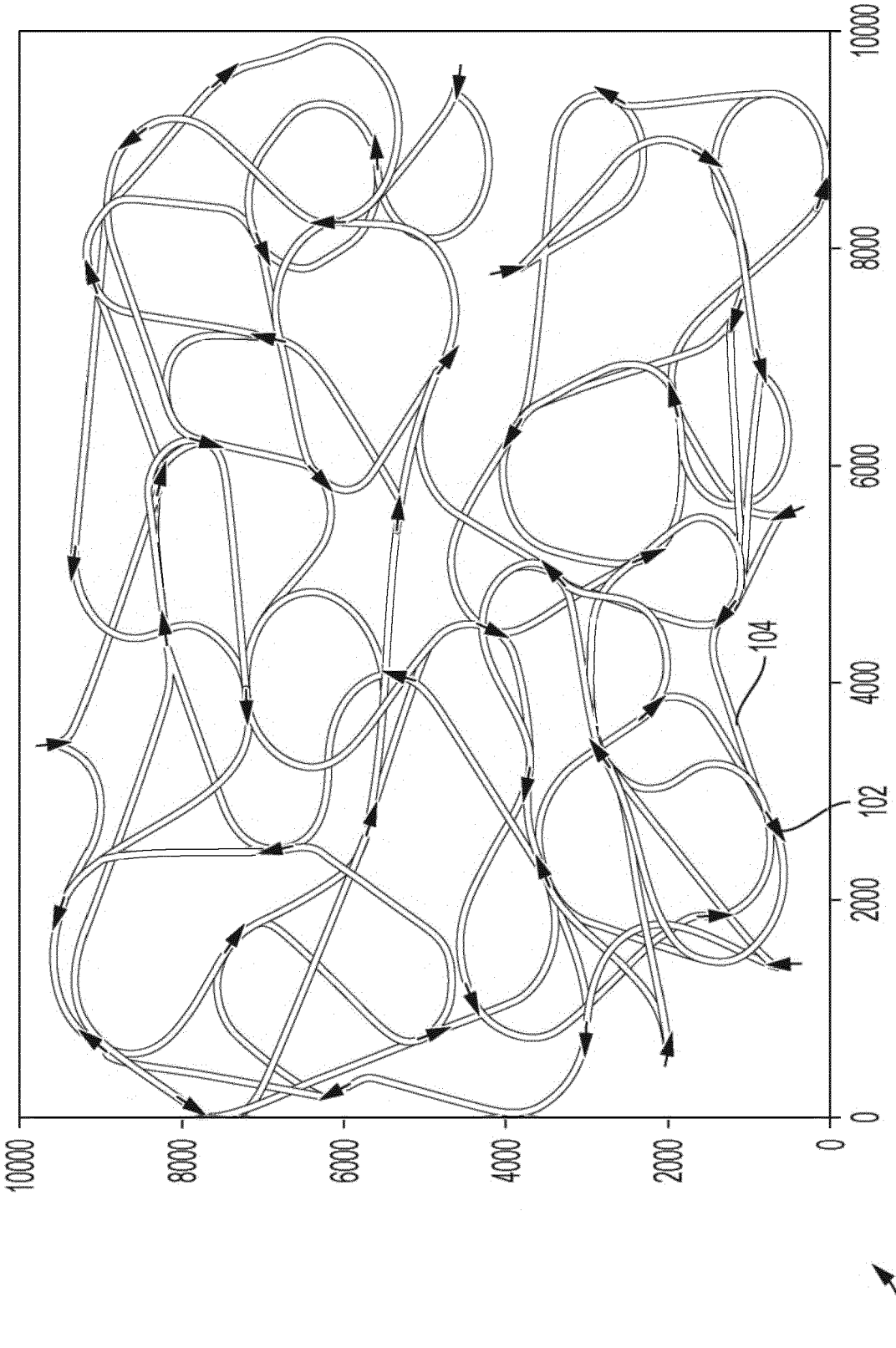
10000
8000
6000
4000
2000
0
0
2000
4000
6000
8000
10000
102
104
100

FIG. 1

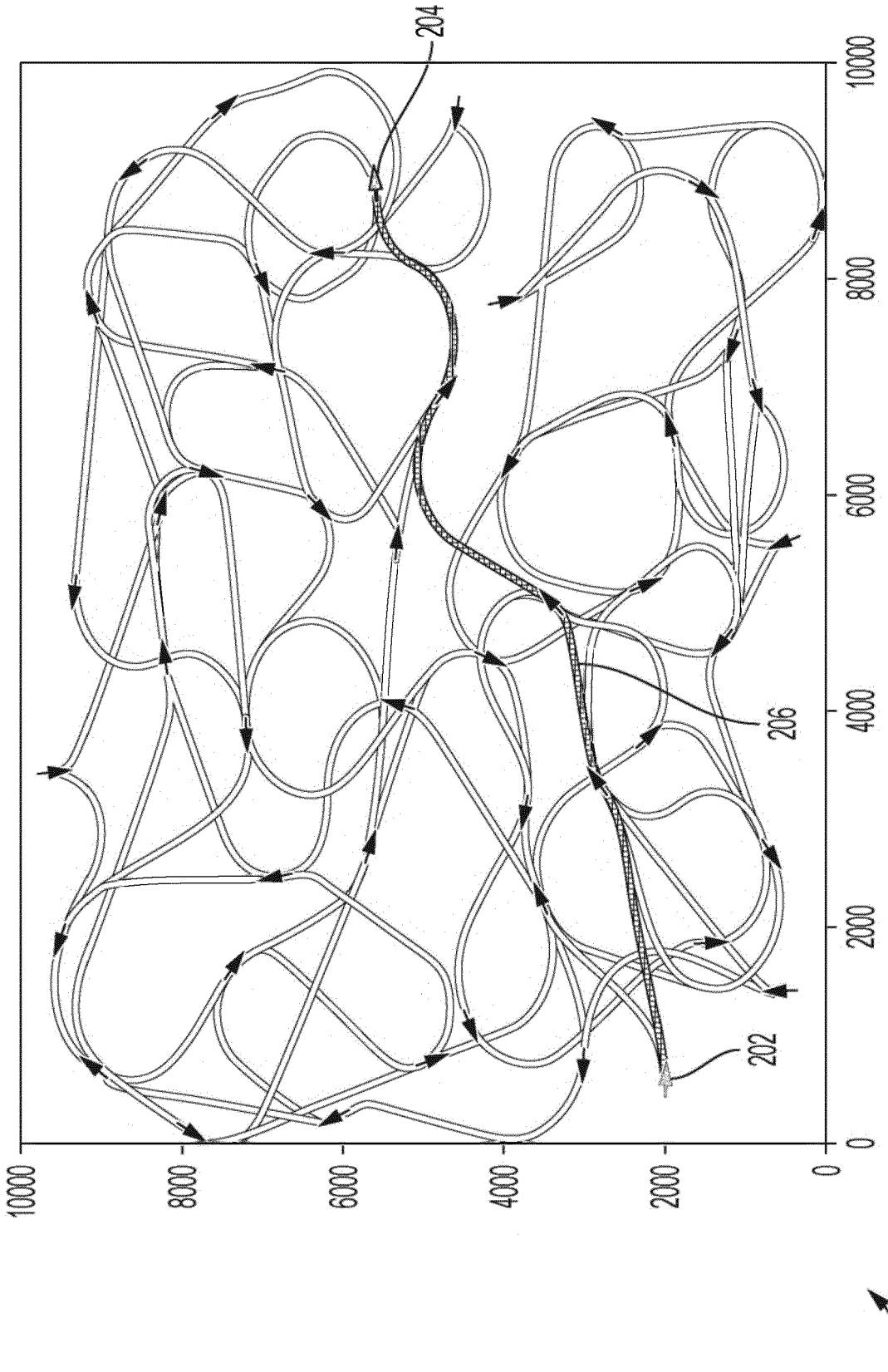

FIG. 2

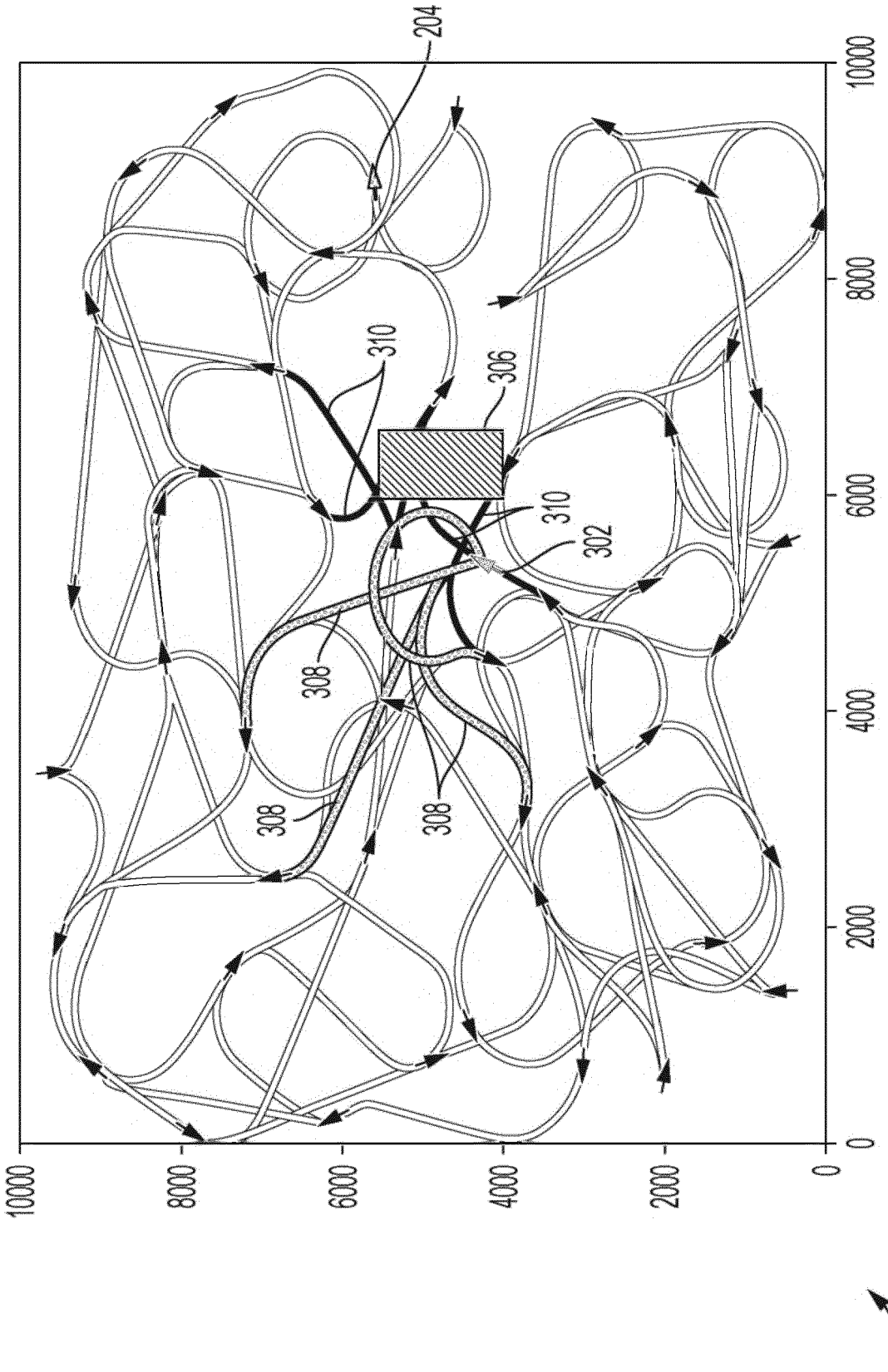
204
310
306
310
302
308
308
308
0
2000
4000
6000
8000
10000
0
2000
4000
6000
8000
10000
300

FIG. 3

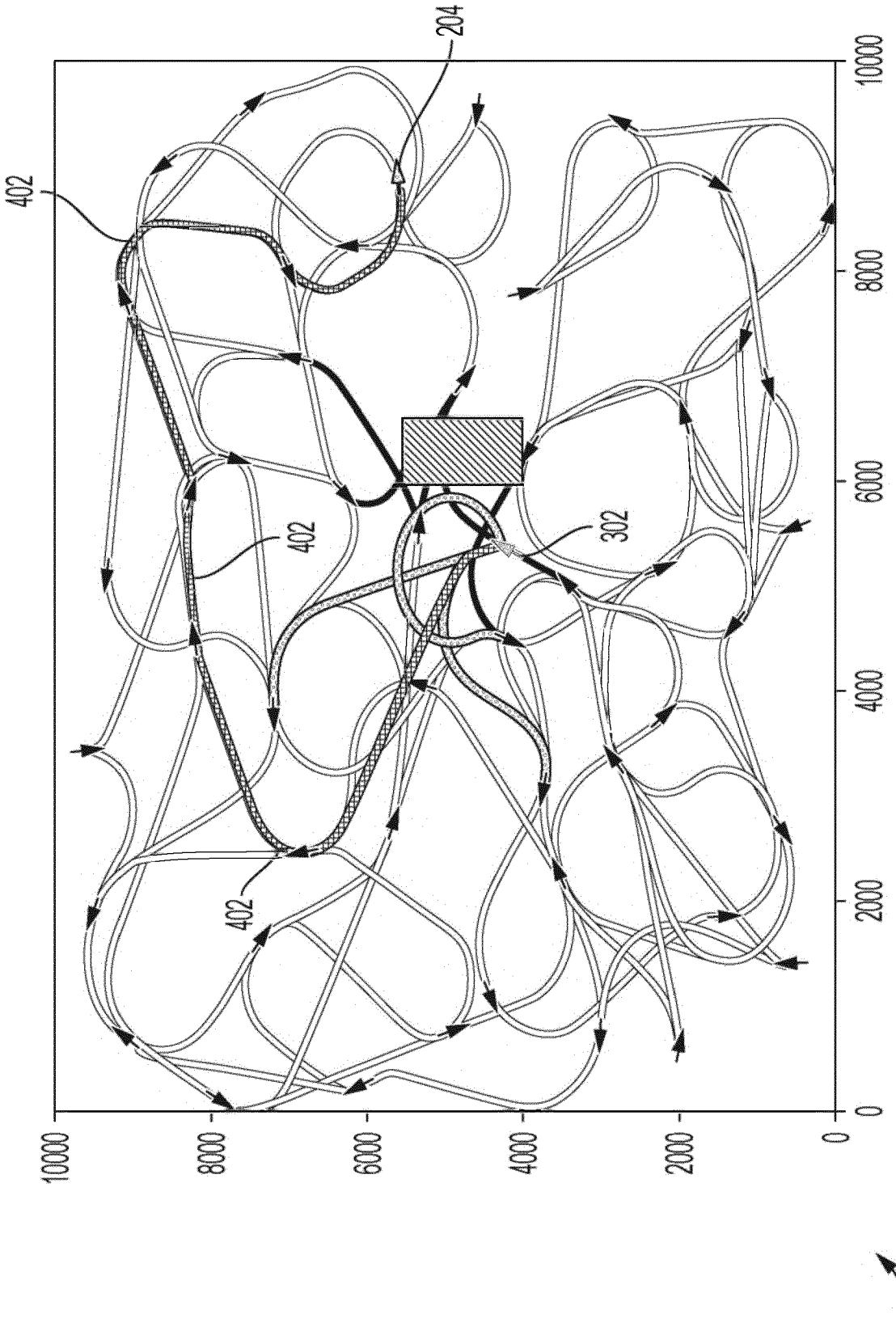
204
402
402
402
302
0
2000
4000
6000
8000
10000
0
2000
4000
6000
8000
10000
400

FIG. 4

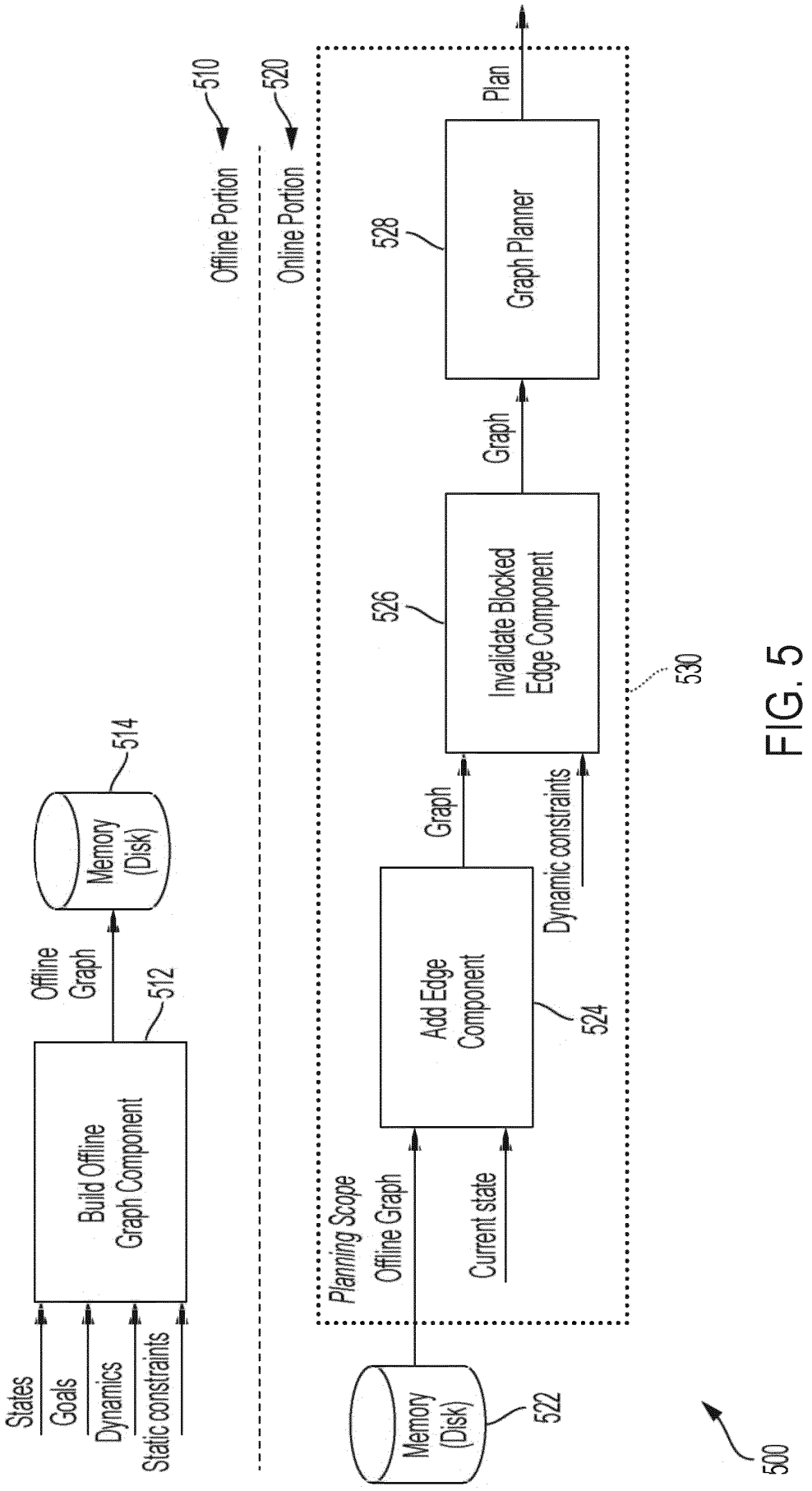
510
520
Offline Portion
Online Portion
States
Goals
Dynamics
Static constraints
Build Offline Graph Component
512
Offline Graph
Memory (Disk)
514
Memory (Disk)
522
Planning Scope
Offline Graph
Current state
Add Edge Component
524
Graph
Dynamic constraints
Invalidate Blocked Edge Component
526
Graph
Graph Planner
528
Plan
530
500

FIG. 5

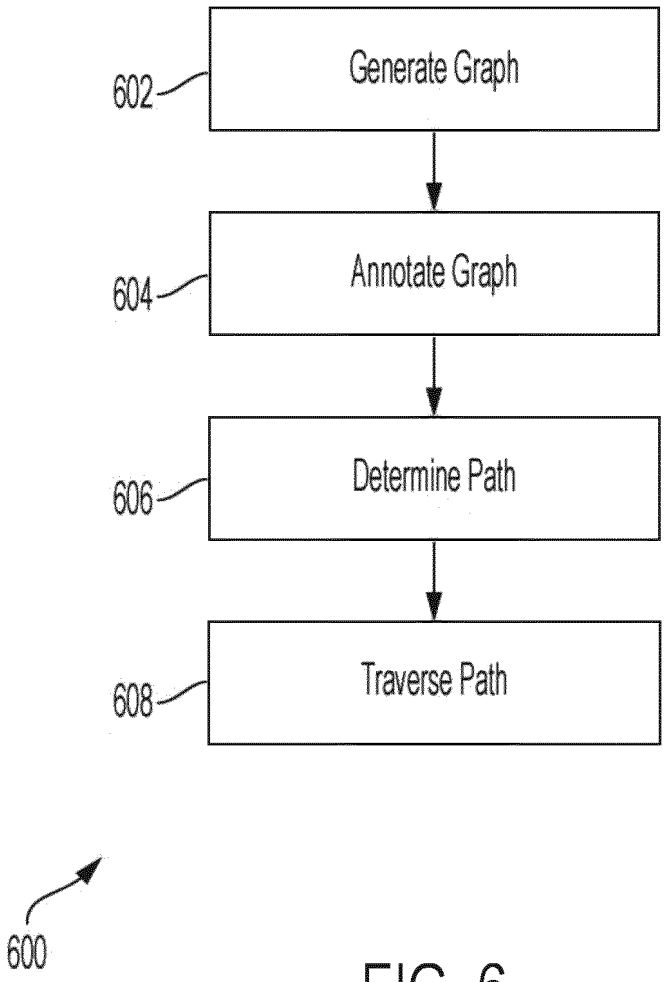
Generate Graph
602
Annotate Graph
604
Determine Path
606
Traverse Path
608
600

FIG. 6

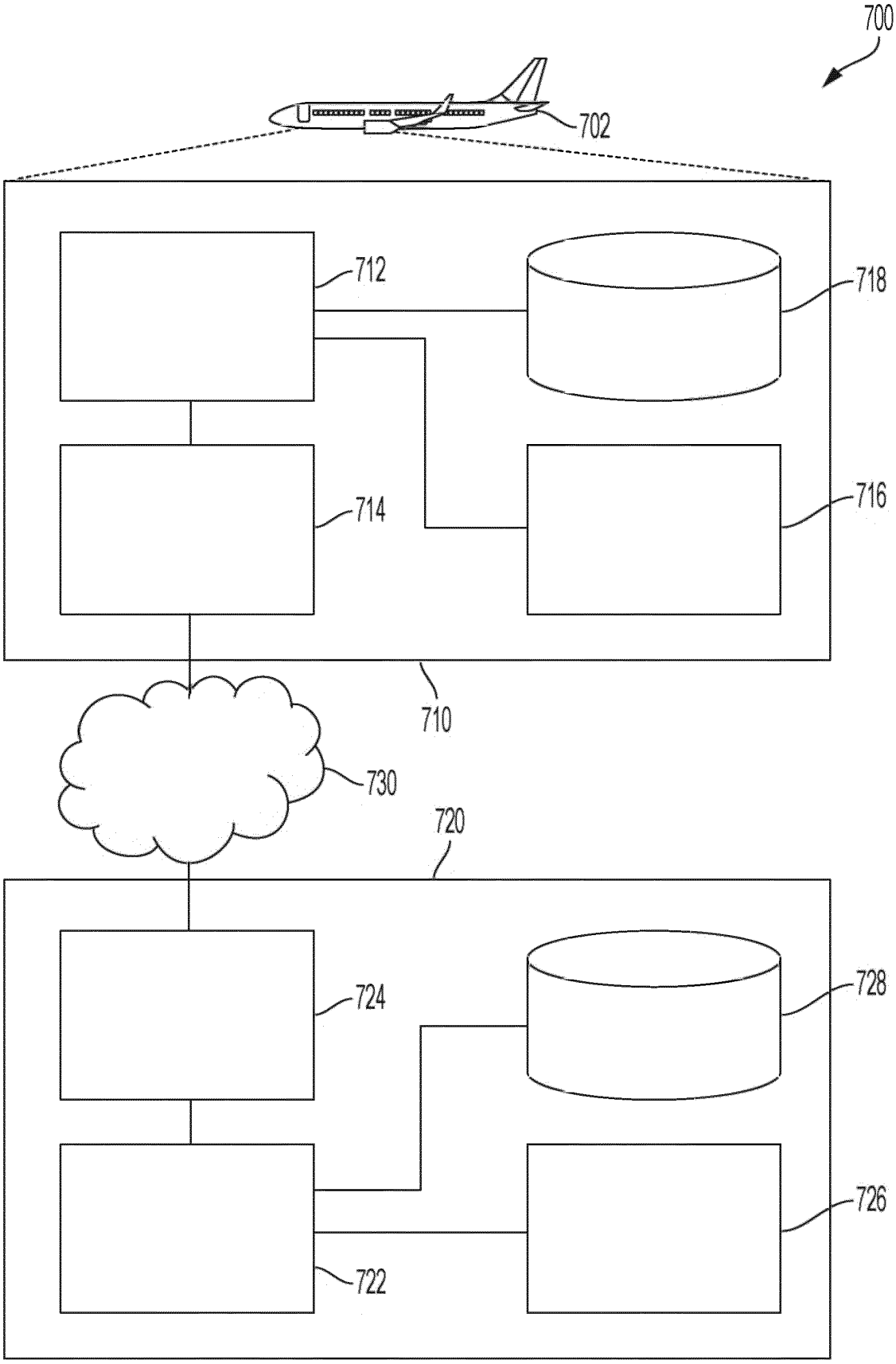

FIG. 7

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 2021134167 A1 **[0004]**
- US 2009210109 A1 **[0005]**